GNUstep

Concrete Architecture

Group 18
URL: https://youtu.be/zoKlk1uXgjo

Group Distribution

- **Lixing Yang**

Group Leader & Abstract & Introduction and Overview & Use case & Lessons Learned

- **Chi Ma**

Presenter & 2nd-level Subsystem Analysis & PowerPoint & Video & Lessons Learned

- **Tiantian Sang**

Presenter & 2nd-level Subsystem Analysis & PowerPoint & Video & Lessons Learned

- **Nick He**

Subsystem Analysis: SCI Engine & Lessons Learned

- **Dunyi Xie**

Top-level Architecture & Lessons Learned

- **Zhiming Jin**

Derivation Process & Lessons Learned

Abstract

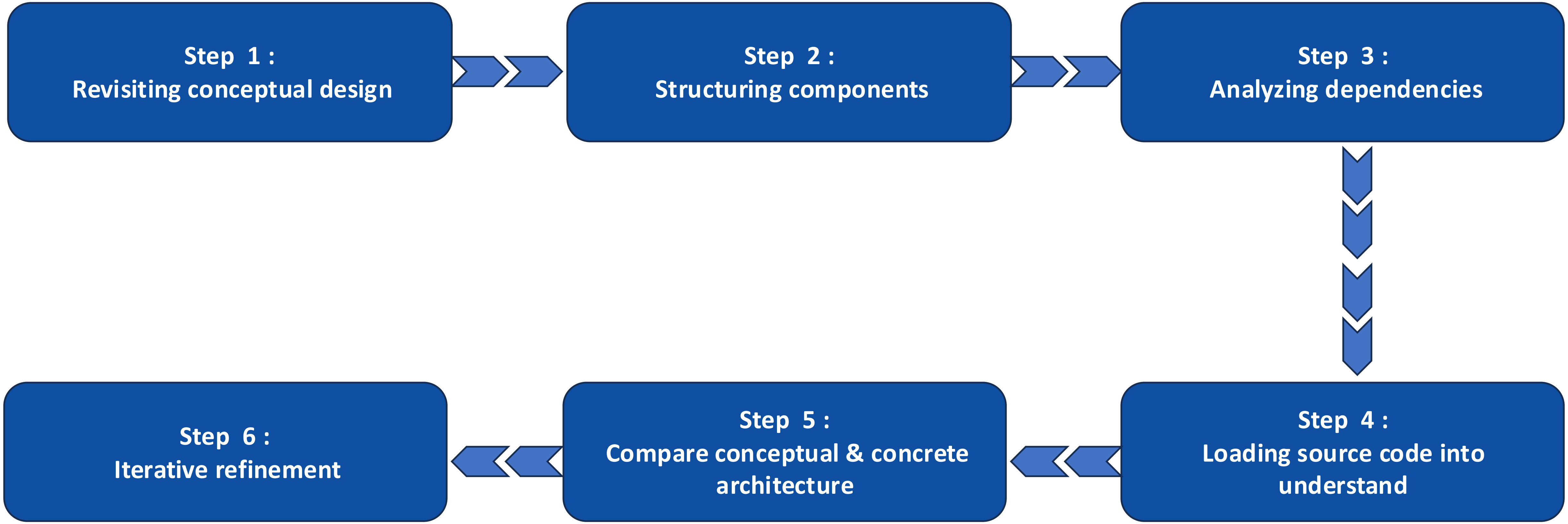
- **Analyzed GNUstep architecture and subsystem interactions**
- **Used Understand tool for component and dependency analysis**
- **Found unexpected dependencies via Reflexion analysis**
- **NSView subsystem diverges from classic MVC structure**
- **Use cases: UI design in Gorm, project setup in Project Center**

Introduction and Overview

The report is structured as :

- Section 1: Abstract
- Section 2: Introduction & Overview
- Section 3: Derivation Process
- Section 4: Top-level Architecture
- Section 5: Subsystem Analysis
- Section 6: 2nd level Subsystem Analysis
- Section 7: Use case
- Section 8: Lesson Learned

Derivation Process



Step 1 :
Revisiting conceptual design

Step 2 :
Structuring components

Step 3 :
Analyzing dependencies

Step 4 :
Loading source code into understand

Step 5 :
Compare conceptual & concrete architecture

Step 6 :
Iterative refinement

Conceptual Architecture

- Layered design: Foundation, GUI, Back

- Foundation: Core data structures, I/O, networking

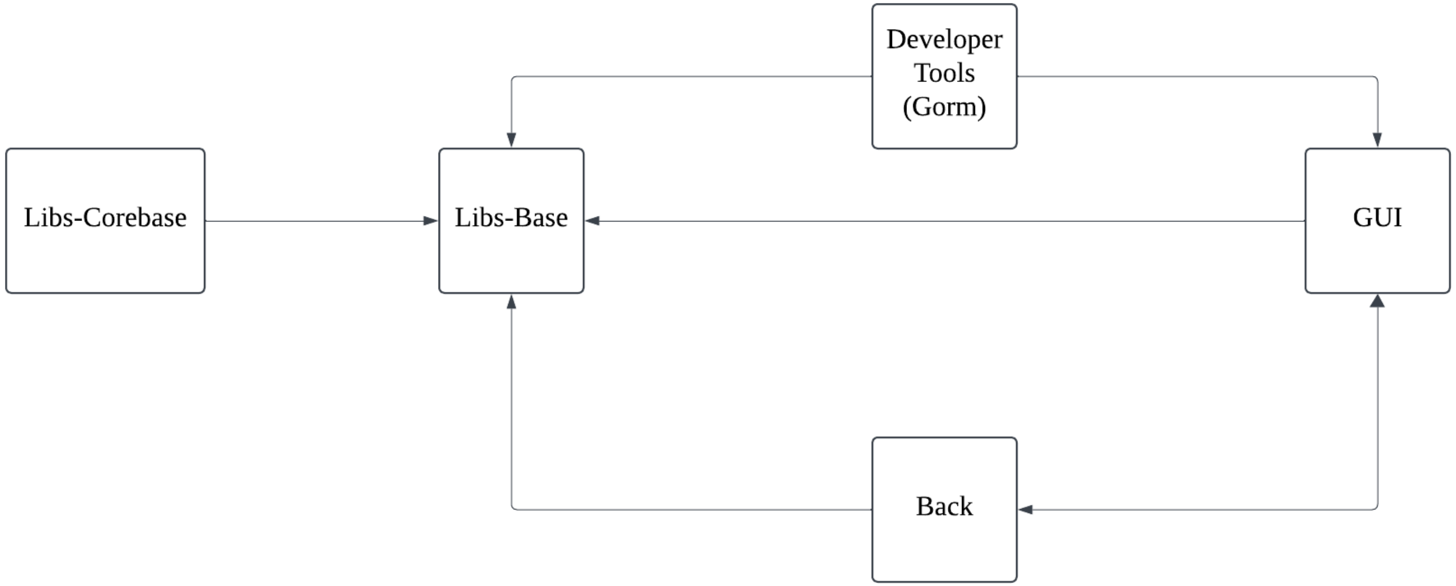
- GUI: Interface elements (windows, views, controls)

- Back: Platform-specific rendering (X11, GDI)

- Developer Tools: Gorm, ProjectCenter, GNUstep Make

- CoreBase: Adds cryptography & advanced data features

Top-level Architecture



Top-level Architecture

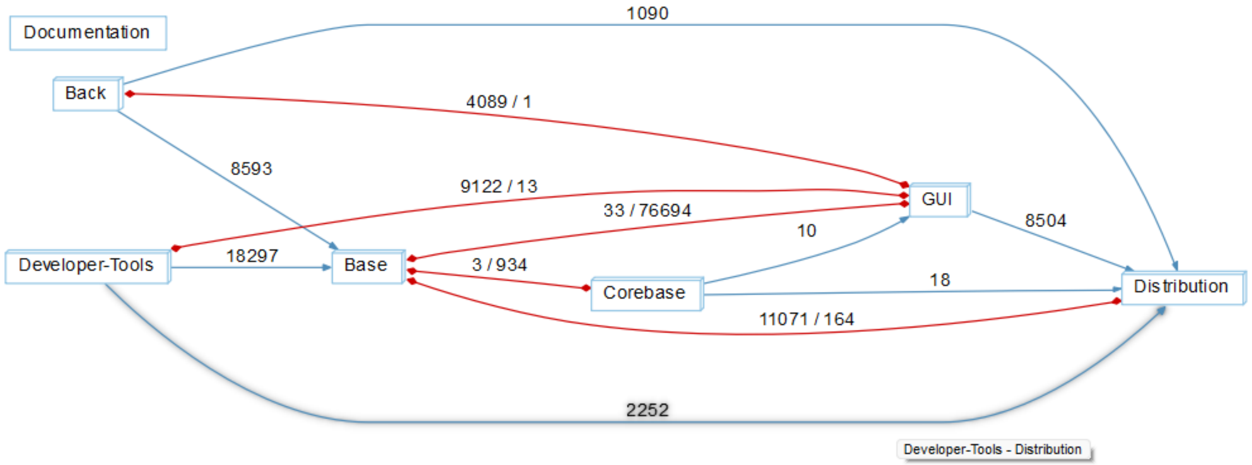
From Understand

- Generated actual architectural views from source code

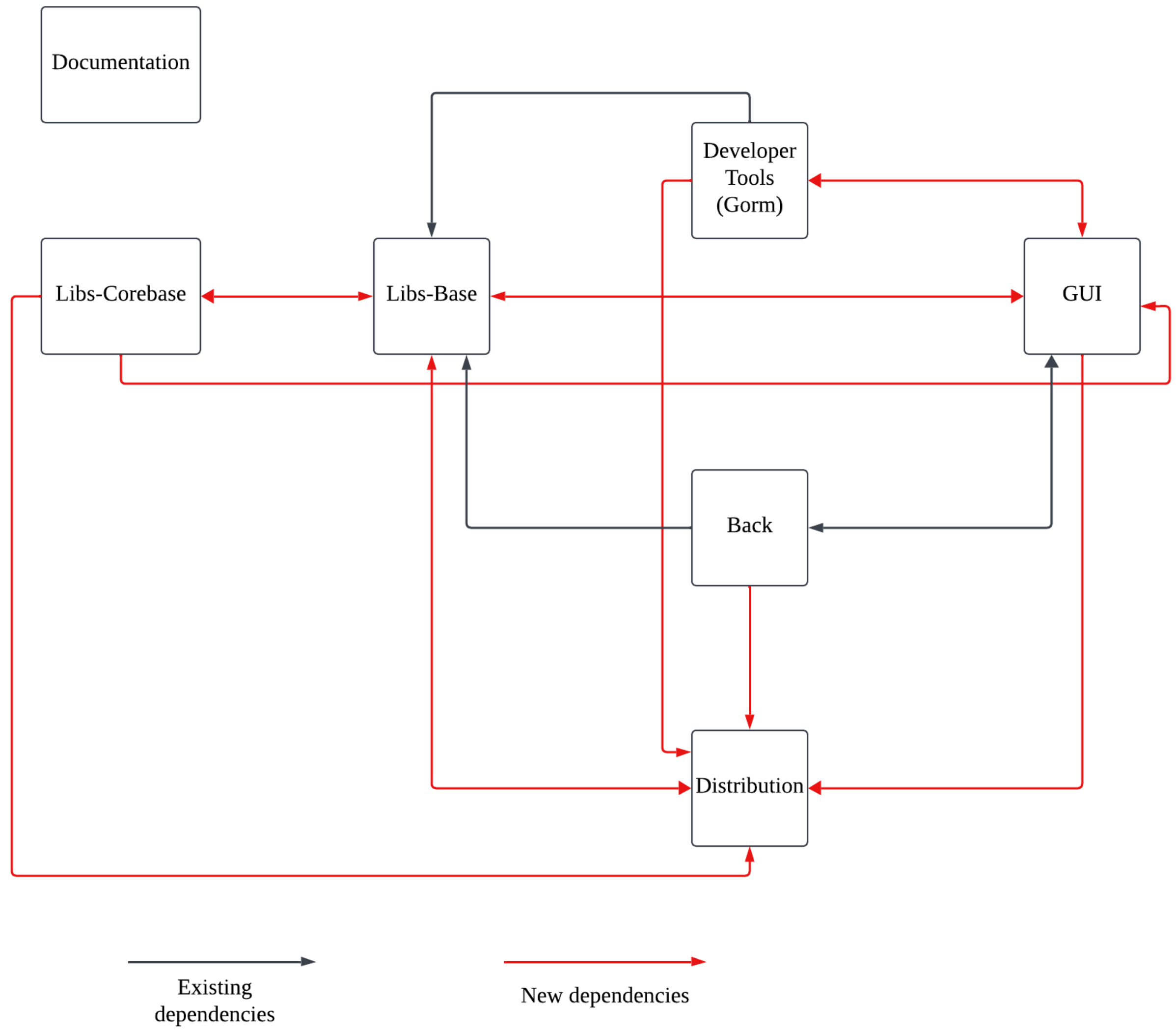
- Revealed unexpected module dependencies

- Identified 3 new components:
  - Documentation: Guides, API references, manuals
  - Distribution: Packaging and deployment support
  - Visual Integration: GUI used in data processes

- Compared conceptual vs. concrete architecture

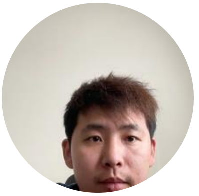


Top-level Architecture

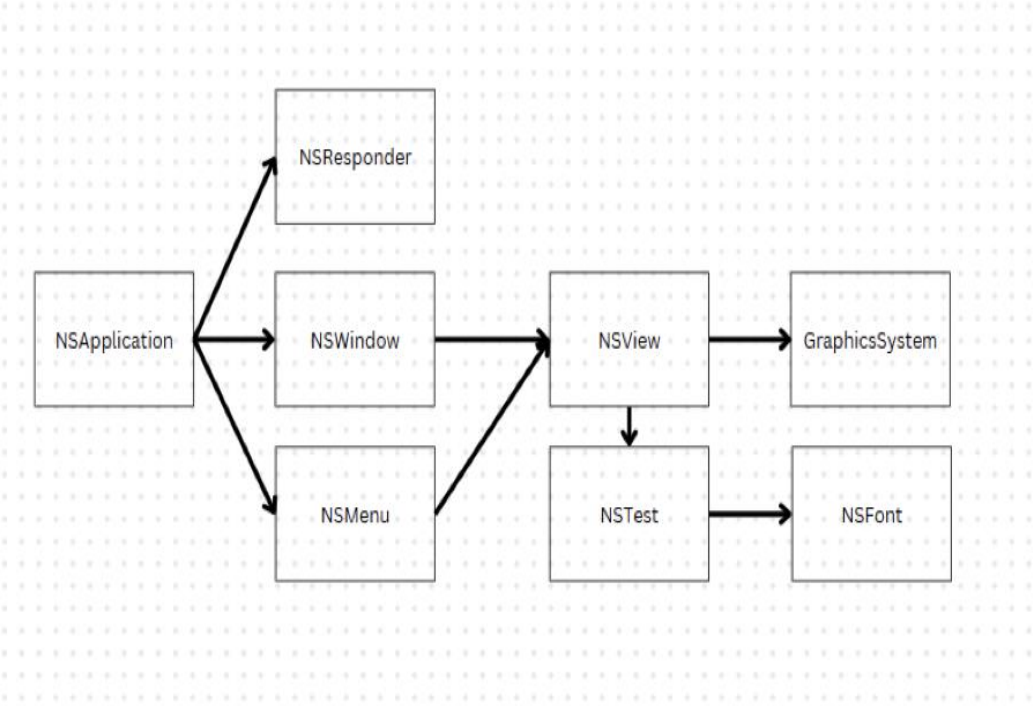


Concrete Architecture

- Base directly calls GUI – breaks expected layering
- CoreBase uses GUI classes for visual feedback
- GUI collaborates with Distribution for packaging UI
- Developer Tools rely on Distribution for deployment
- Base and Distribution share metadata and resource handling
- CoreBase and Base share advanced cryptographic workflows



Subsystem Analysis: GUI



Conceptual Architecture

- Simplified structure with clear responsibilities.

- Direct dependencies between UI and graphic.

- Centralized event handling via NSResponder.

- Assumed interactions:

NSApplication → NSWindow → NSView

NSResponder manag

GUI Concrete Architecture

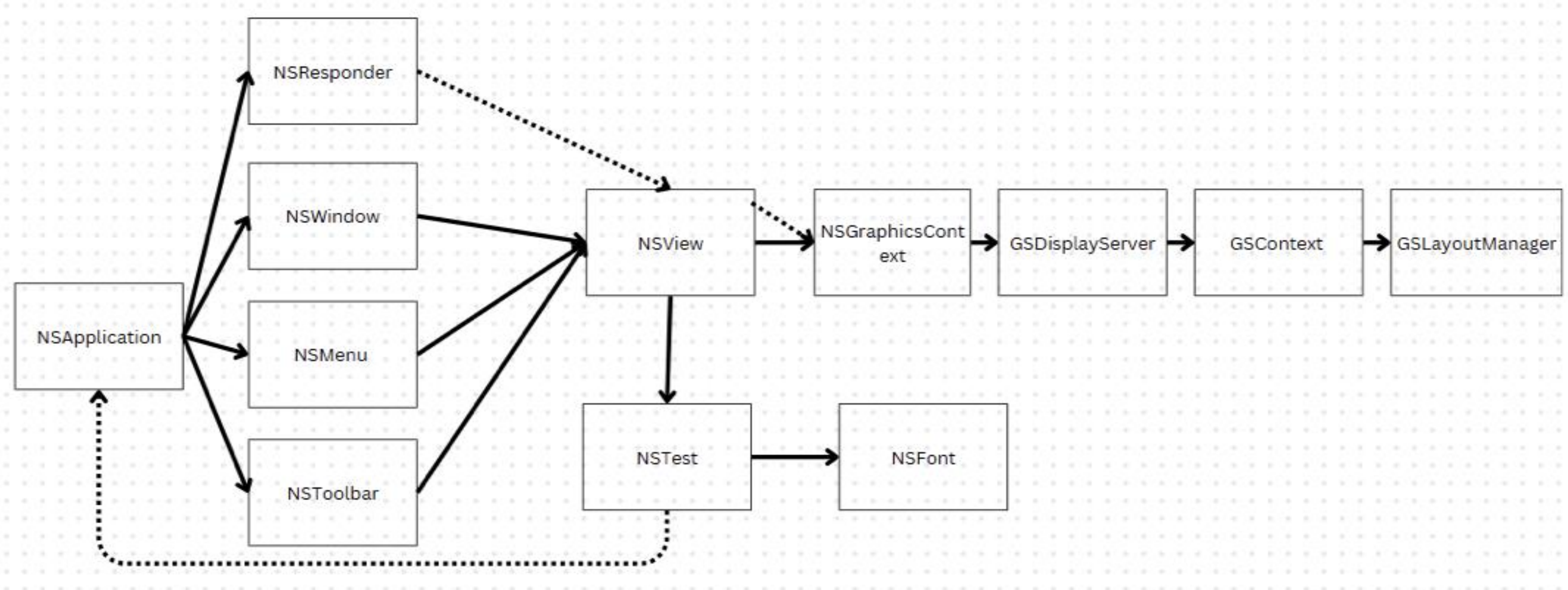
- More layers added for flexibility and performance

- NSGraphicsContext between NSView and rendering

- GSDisplayServer, GSContext, and GSLayoutManager handle UI rendering.

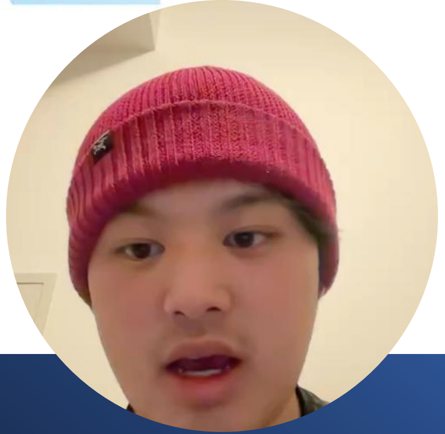
- Event handling is more distributed than expected.

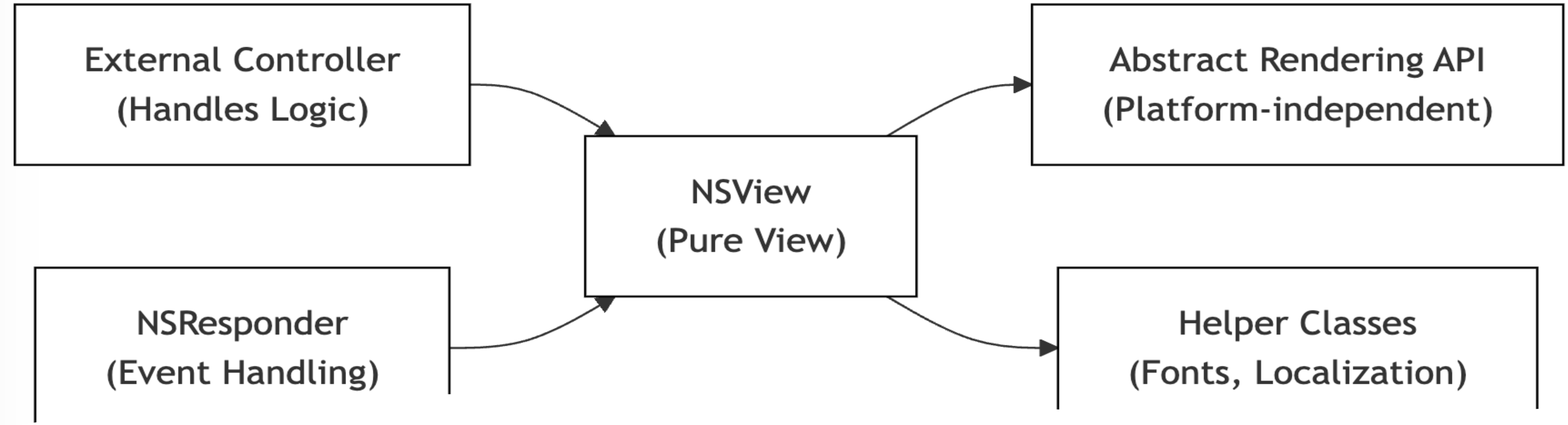
- Increased complexity but optimized for efficiency.



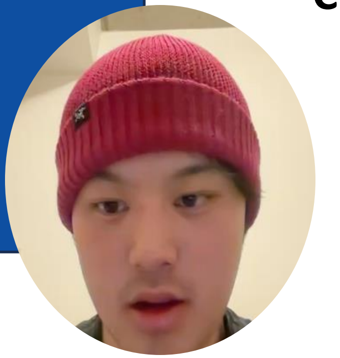
Reflection Analysis

- **More abstraction layers** improve modularity & performance.
- **Increased dependencies** optimize event handling & rendering.
- **Bidirectional communication** improves efficiency but increases complexity.
- Some assumed dependencies were missing, while unexpected ones were introduced.





Second Level Subsystem: NSview

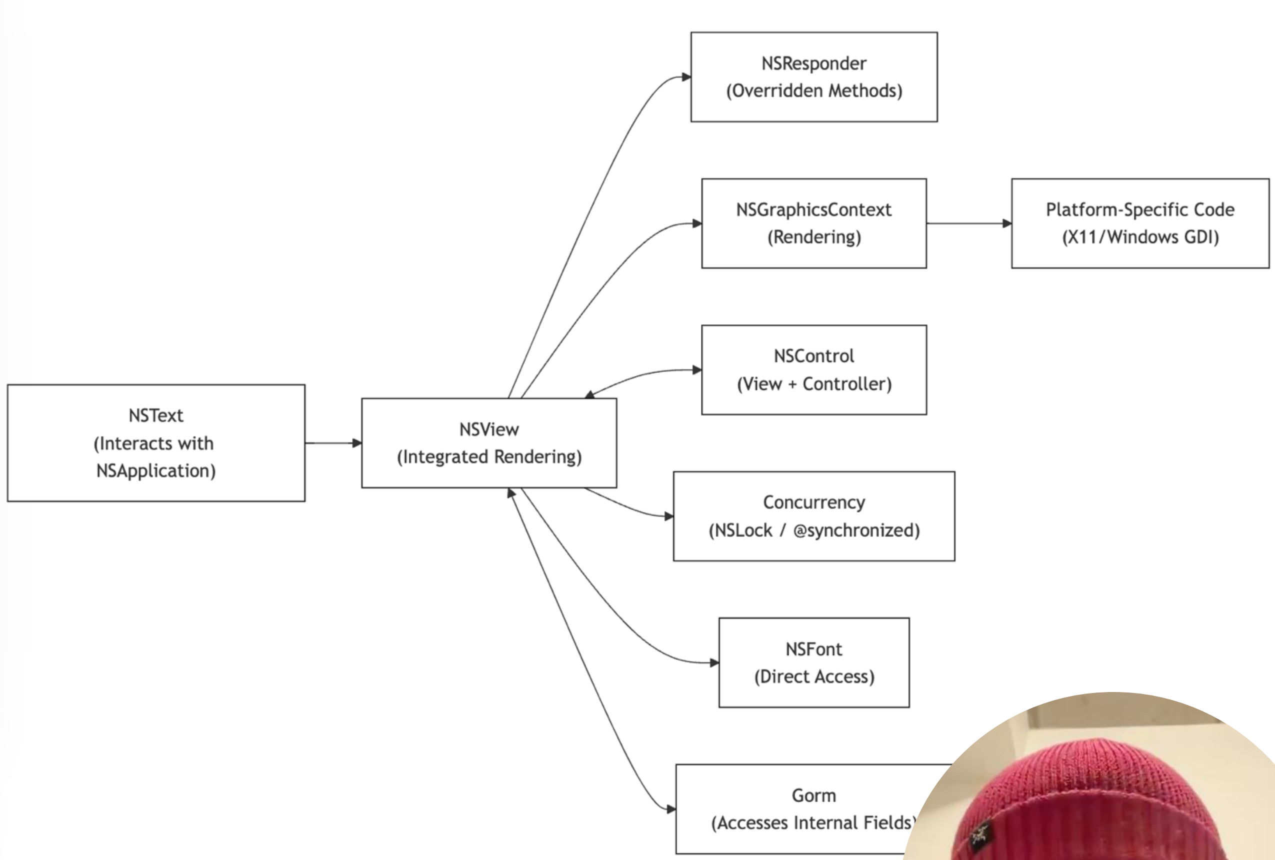


- Visual content, layout, and rendering; User interaction
- Positioned between NSWindow (container) and the rendering back-end.
- **Conceptual View:**
  - Expected to follow strict MVC principles (acts only as a View).
  - Uses an abstract rendering system for platform independence.
  - Handles event processing through NSResponder, centralized input management.

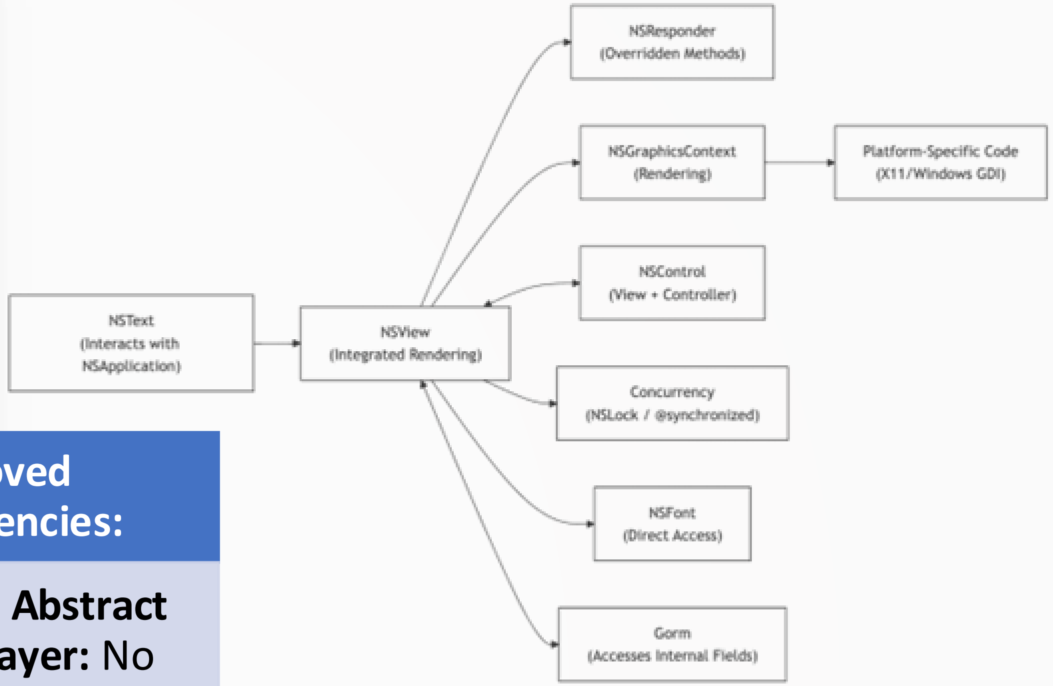
NSView Concrete Architecture

- **NSView**: directly communicates with NSGraphicsContext (no abstraction layer).
- **NSControl:** Combines view and control logic, a bidirectional relationship.
- **Concurrency:** NSView spawns threads and uses locks.
- **NSFont:** NSView directly pulls font information.
- **Gorm:** Accesses internal fields of NSView, bypassing public APIs and creating tighter coupling.
- **NSResponder:** NSView subclasses override event handling, mixing view and responder responsibilities.

- Illustrates increased coupling and reduced modularity.



Reflection Analysis


NSResponder (Overridden Methods)
NSGraphicsContext (Rendering)
Platform-Specific Code (X11/Windows GDI)
NSControl (View + Controller)
NSText (Interacts with NSApplication)
NSView (Integrated Rendering)
Concurrency (NSLock / @synchronized)
NSFont (Direct Access)
Gorm (Accesses Internal Fields)

Missing Dependencies:

- **NSView → NSGraphicsContext**: Expected abstract graphics layer is absent.
- **NSView → NSFont**: NSView directly accesses NSFont instead of using a helper layer.
- **NSText → NSApplication**

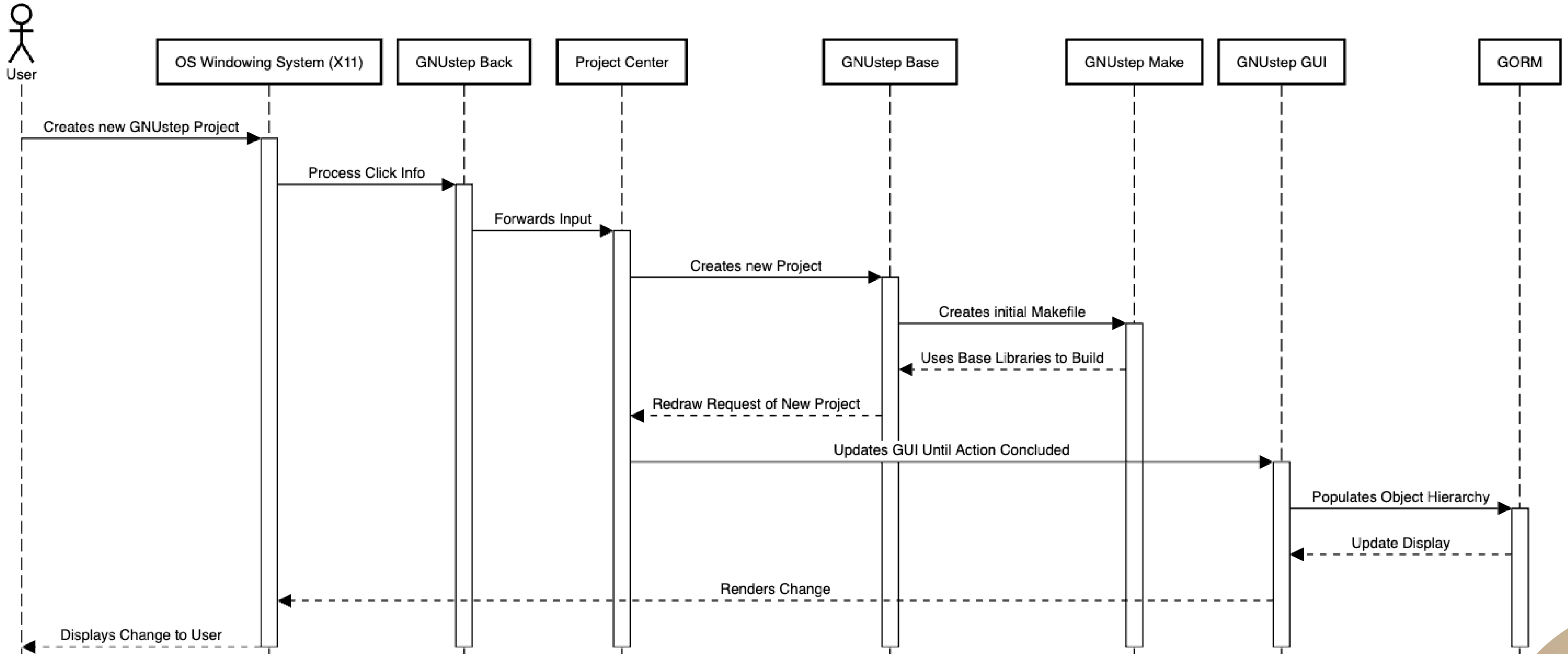
Unexpected Dependencies:

- **NSView → Platform-specific code**
- **NSControl → Controller Logic:** NSControl integrates controller logic.
- **Gorm → Internal Fields of NSView:** Gorm directly accesses internal NSView fields, reducing modularity.

Removed Dependencies:

- **NSView → Abstract Graphics Layer:** No separate abstract graphics layer for rendering.
- **NSView → External Controller**: External controllers are sometimes bypassed in interactive vie

Use Case 1:



User

OS Windowing System (X11)

GNUstep Back

Project Center

GNUstep Base

GNUstep Make

GNUstep GUI

GORM

Creates new GNUstep Project

Process Click Info

Forwards Input

Creates new Project

Creates initial Makefile

Uses Base Libraries to Build

Redraw Request of New Project

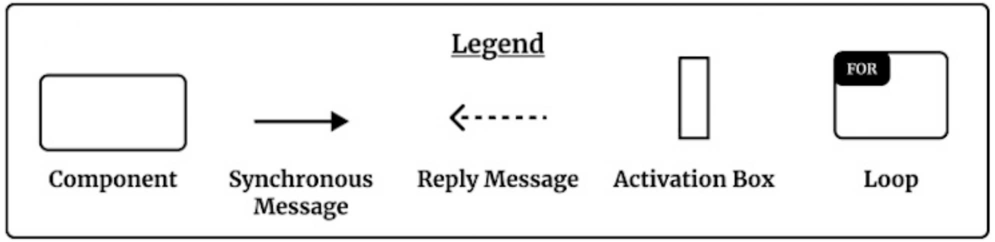
Updates GUI Until Action Concluded

Populates Object Hierarchy

Update Display

Renders Change

Displays Change to User



Legend

Component

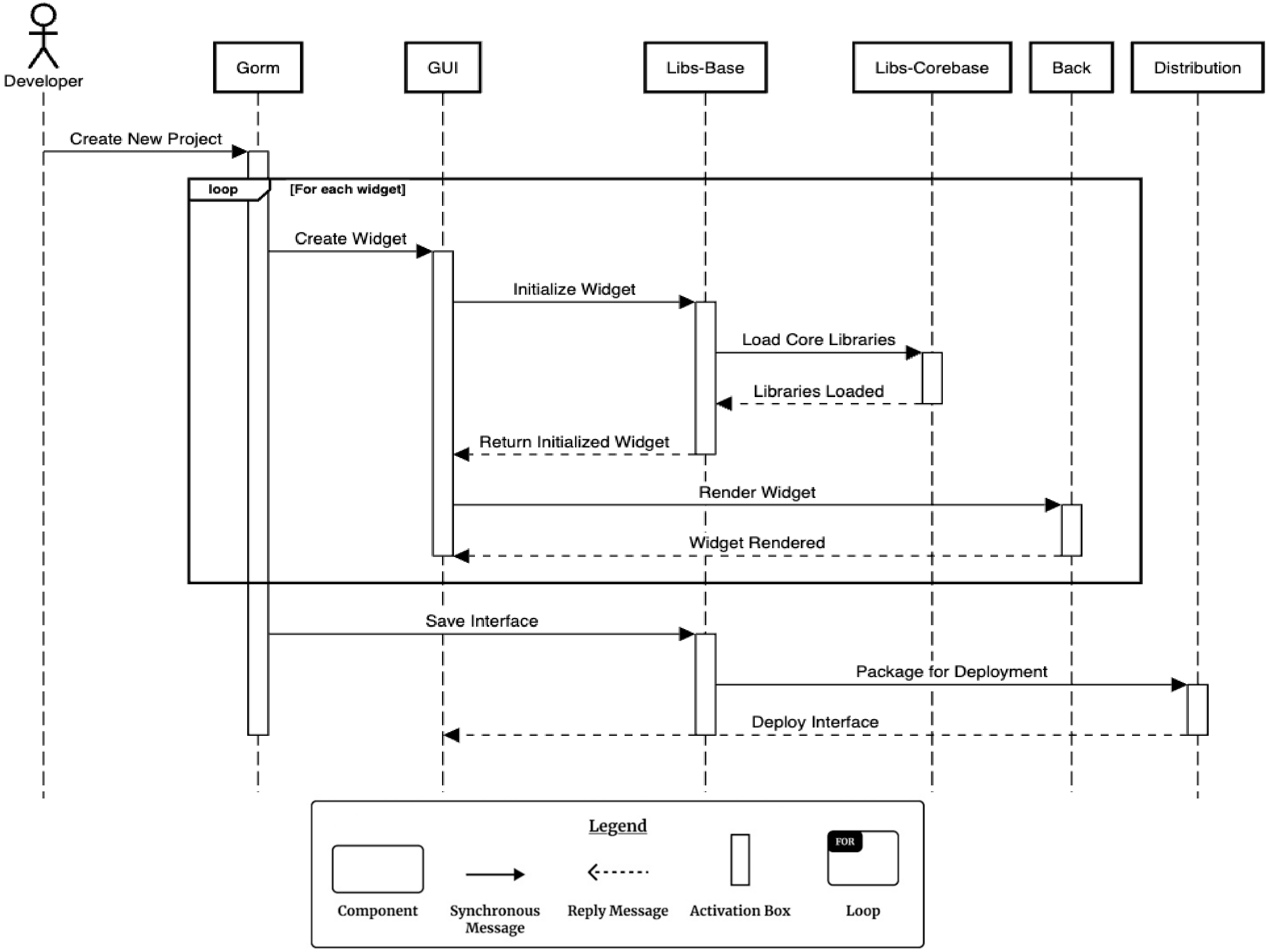
Synchronous Message

Reply Message

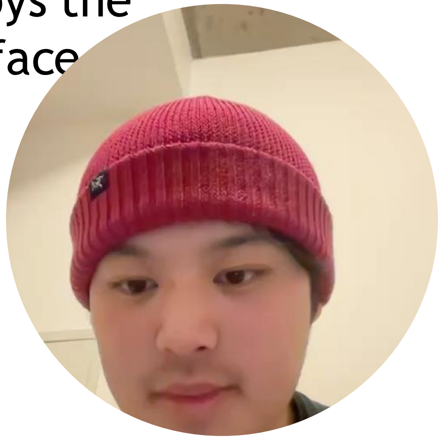
Activation Box

Loop

Use Case 2:



- System enters widget creation loop.
- Gorm requests GUI to create widgets.
- GUI initializes widgets via Libs-Base.
- GNUstep Back handles rendering.
- Gorm saves and deploys the interface



Lesson learned

Conceptual vs Concrete Architecture

NSView

Concurrency

Coupling & Modularity

Practical Constraints

Reference

- GNUstep. GUI Library Reference Manual. Retrieved from
- https://www.gnustep.org/resources/documentation/Developer/Gui/Reference/
-

- GNUstep. GNUstep Developer Documentation. Retrieved from
- https://www.gnustep.org/developers/documentation.html
-

- GNUstep. GNUstep-gui Source Code Repository. Retrieved from
- https://github.com/gnustep/libs-gui